Sustainability of Public Policy

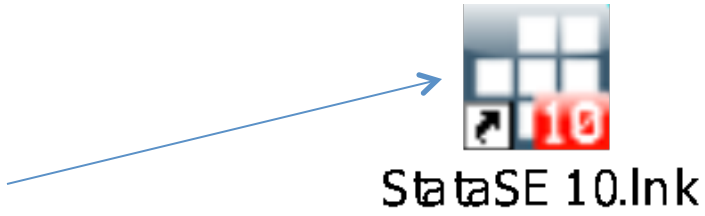
Lecture 1

Introduction STATA

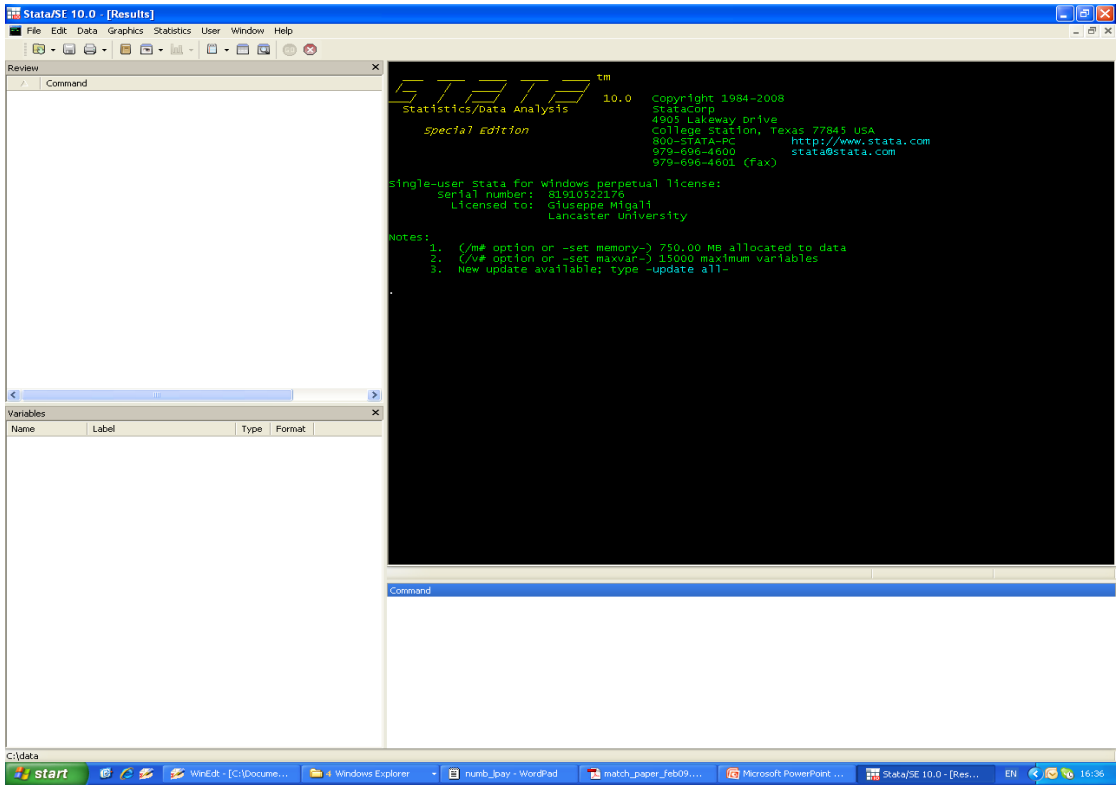
Rossella Iraci Capuccinello

Getting started in STATA

- Start STATA
  - Simply click on icon



  - Stata should look like this:
    - Buttons/menu
    - Review window
    - Results window
    - Command entry window
    - Variables window

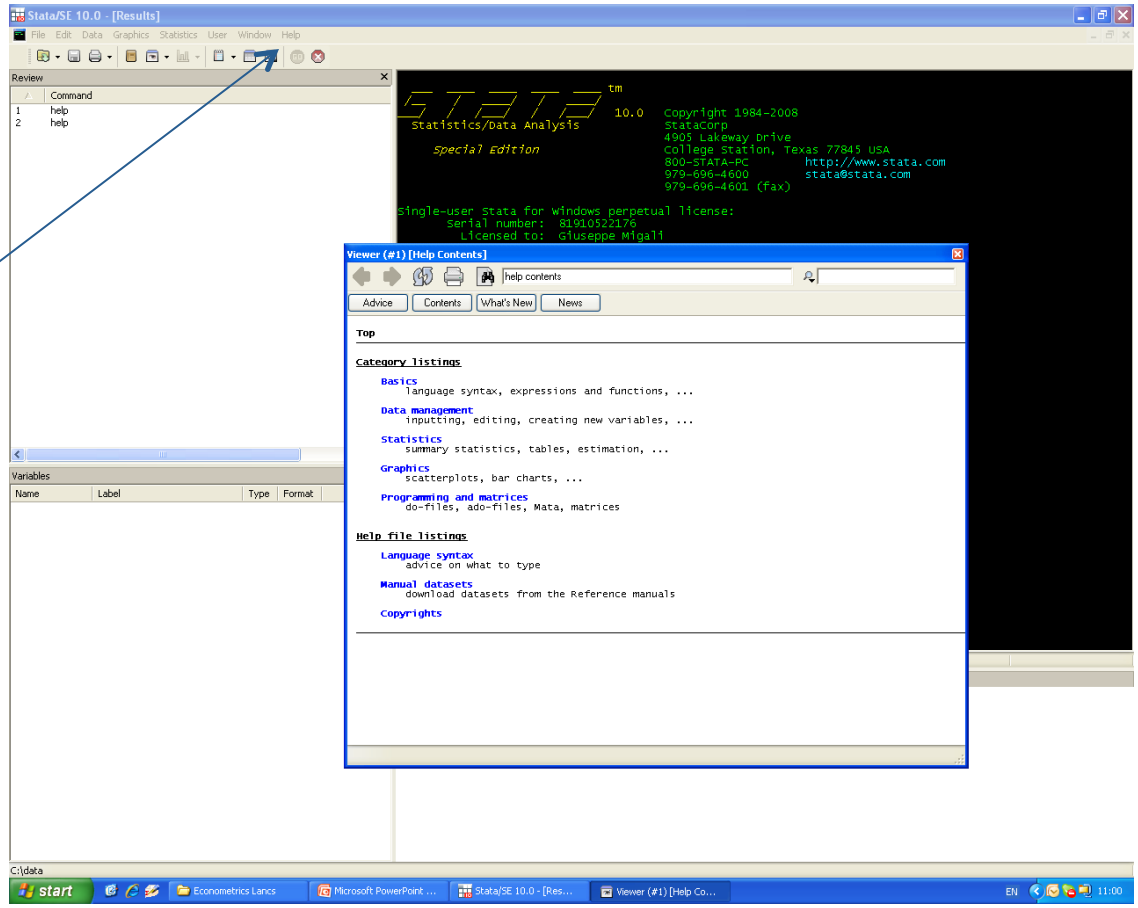


- To exit type:

  exit

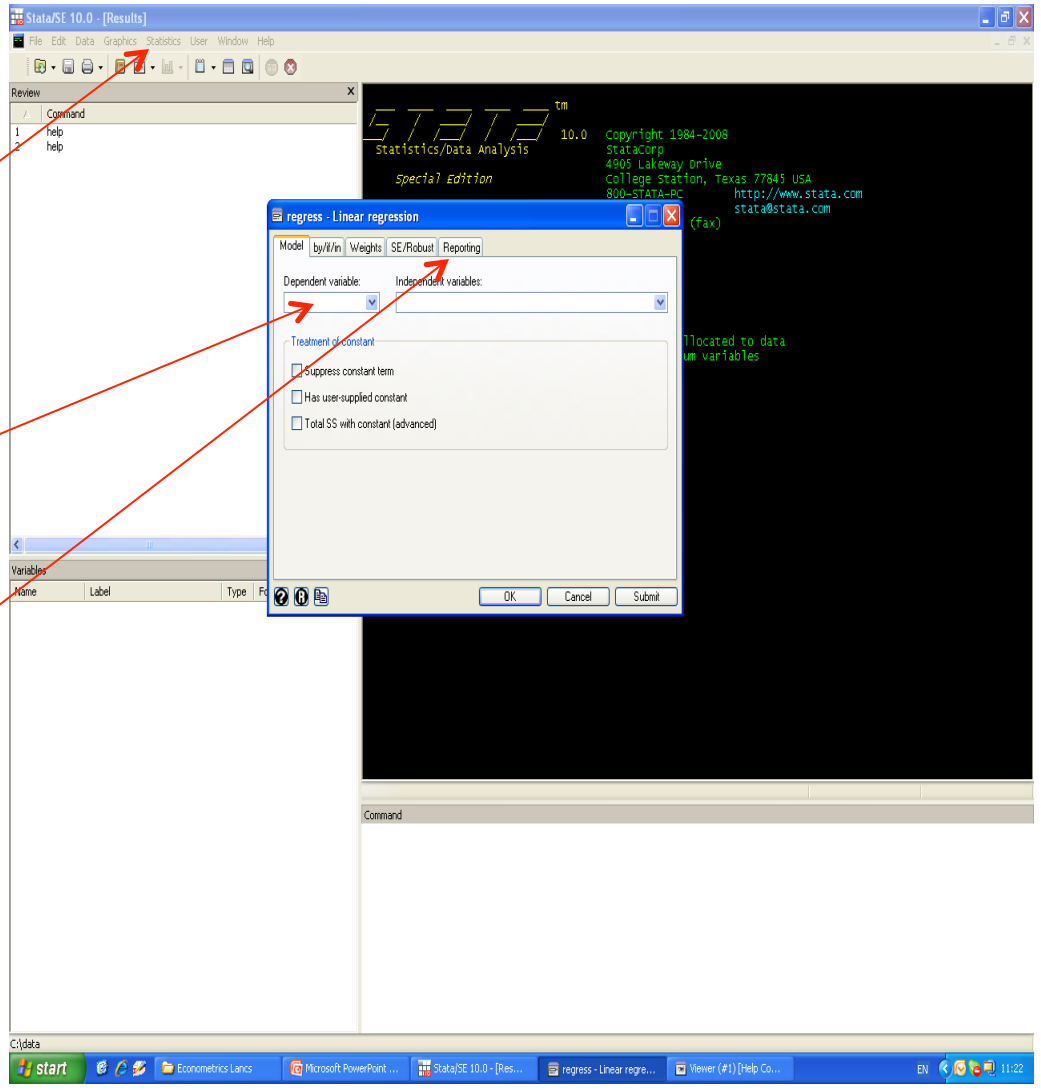
Getting help

- Lots of on-line help
- Click on *help* on menu
- Type *help xxx* for help on the “xxx” command



Click and point in v10

- Use the menu bar to click and point to most commands
- Then fill in the boxes in the resulting dialog box
- Click on tabs for further options



Important features

- NOTE
  - Always use lowercase in STATA
  - ren *, lower
- *More*
  - When you see *--more--* press the spacebar! Or type set more off
- *Break*
  - To stop scrolling output, hit the red cross (Ctrl-Break)
- Not enough memory
  - *set mem XXXm* (resizes to allow XXX mb)
  - *set matsize XXX* (max matrix to XXX square)
- Type set to check all Stata settings

Loading up some data

- You will usually want to open some dataset
  - *Stata* expects datasets to be rectangular with columns being variables and rows being obs
- Several ways of getting data into STATA:

use myfile (or click *file open* on the menu bar)

(opens a stata format file called myfile.dta)

use var1 var2 var3 using myfile in 1/1000 if var4==1

(loads var1, var2, var3 for the first 1000 obs if var4=1)

insheet using myfile.csv (or .txt)

imports *csv* file which Excel can read (or "*text*" file)

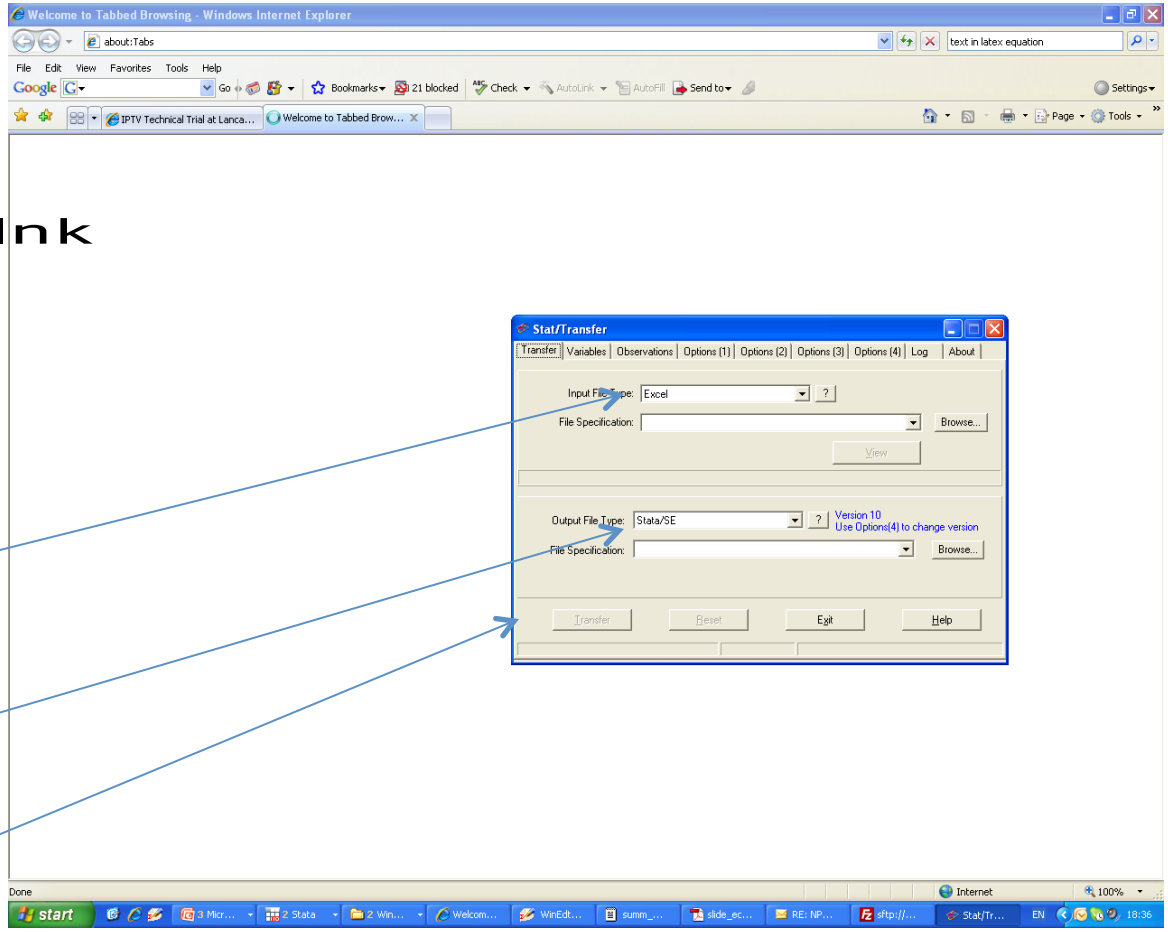
clear clear memory

Data

- Huge variety of datasets from data archives
  - UKDA , ICPSR, and around the world:
    - www.esds.ac.uk/international/access/map.asp
  - Research centres
    - NBER www.nber.org
  - Government depts and international agencies
    - US Census Bureau and World Bank
  - And journal websites (like AER)
    - Most major journals make datasets and code easily available

Stat-Transfer

- Use STAT-TRANSFER to convert data.
- Click on StatTransfer.lnk
- Stat-transfer is “point and click”.
- Just tell it the file name and format
- and the format you want it in.
- Click “transfer”.



Practicising

- Import Stata's own demos *sysuse*
  - E.g. *sysuse auto*
- Many datasets available at specific sites
  - E.g. STATA's own site has all the demo data
- Use the *webuse* command to load the files directly into stata without copying locally

  webuse auto /* gets *auto* from www.stata.com*/

Web resources

- STATA is web-aware
  - *update* /*updates v10.0 from www.stata.com */
- Statalist is an email listserv discussion group
- *The Stata Journal* is a refereed journal
- SSC Boston College STATA Archive
  - Files can be downloaded in Stata using *ssc*
    - E.g. *ssc install outreg2*

    Installs the *outreg2* ado file that makes tables pretty
    - findit finds ado files from the web
    - lookfor finds variables in your data set searching for strings among varnames and labels

Keeping track of output

- STATA allows you to scroll back your screen
- But better to open a log file: it saves commands as well as output
- Click on *file, log, begin* . Or type

  log using myoutput

  my commands……

  list age

  log close

- You can turn logging off and back on

  log off then log on when ready to resume

  log command allows the *replace* and *append*.

Saving output

- Default is a *.smcl* file extension (to "view")

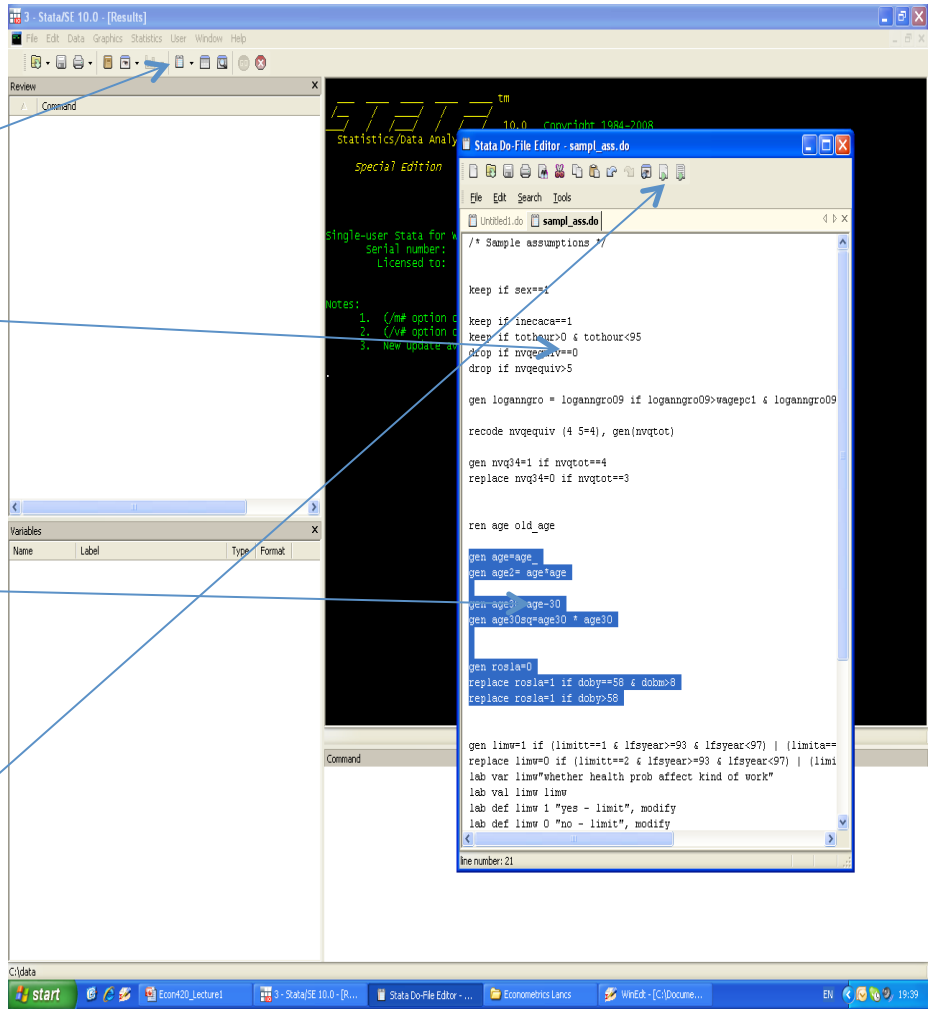
- You might prefer to give your own, say, *.log, .txt*
  - then you get an ASCII file that anything can edit
  - you can *translate* files to and from *.smcl* format
    - click on *file, log, translate* and fill in the dialog box

- It doesn't save graphs
  - Copy graphs (use cut and paste or use menus)

- *Cd* or *pwd* show the current directory, to change it type
  - cd"C:\Documents\Course Ecmtrx\myfolder"
  - ls show the files in the directory

STATA's command file editor

- STATA has an editor that allows you to create *do* files
  - Enter cmds – 1 per line
  - Save the commands in a "*do*" file
  - Highlight commands and click the button with page (or page with text) and right arrow to "run" (or "do") commands.



Command *.do* files

- More complicated ideas can be implemented as a sequence of commands. For example:

  list var1 var2 var3 in 1/10 if var4>=0

  – Lists the first 10 rows of var1 to var3 for which var4≥0

- Collect commands in the editor, save *.do* file.

- Then type *do mycommands.do, nostop*
  – echoes to screen, and keeps going after error
- Or *run mycommands.do* executes "silently"

Saving commands

- It is ALWAYS good practice to use a *.do* file
  - – easy to develop ideas and correct mistakes
- Logging your output is a good way of developing a *.do* file
  - since it saves the commands as well as output

- Or you can just log the commands
  - Type *cmdlog using* xxx.txt

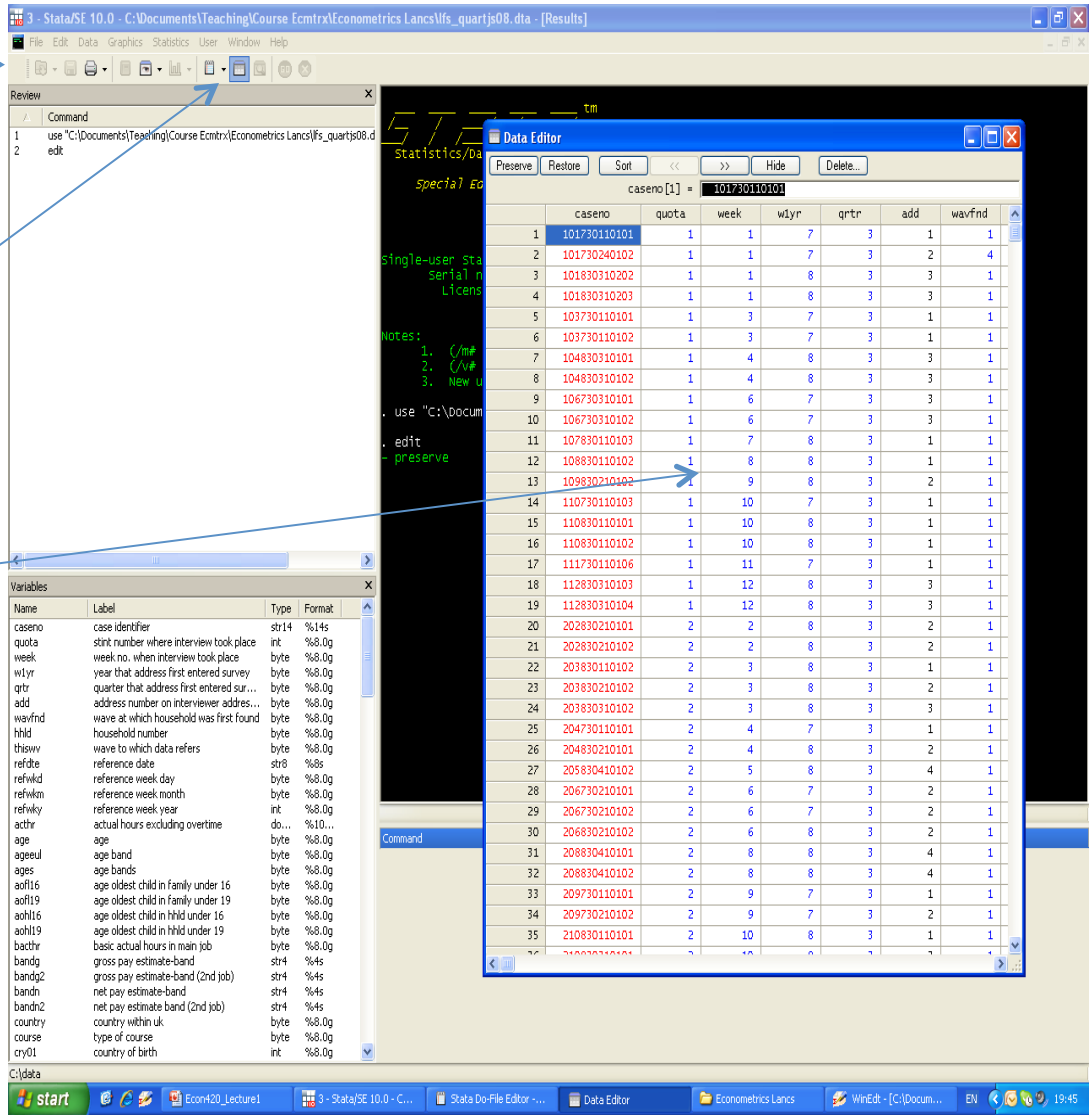
- Then save the file for subsequent analysis

  save newfile

  save, replace take care – it overwrites

Using the data editor

- Open a datafile (eg.lfs_3rdquart08.dta)
- Click on the icon
- Or type *edit*
- You can edit datapoints!
- Or just browse the datafile



Basic data reporting

- *describe (or press F3 key)*
  - Lists the variable names and labels

  describe using myfile
  - Lists the variable names etc WITHOUT loading the data into memory (useful if the data is too big to fit)
- *codebook* (you can also use *inspect*)
  - Tells you about the means, labels, missing values etc

First look at the data

- *summ x1 x2* (or *summarize* or *sum* or *su*)
  - Gives you the means, std devs etc for *x1* and *x2*
- *corr x1 x2 in 1/100 if x4<0 (,cov)*
  - correlation coeffs (or covariances) for selected data
  - *pwcorr x1 x2 x3* does all pairwise corr coeffs
- *tab x1 x2* (or *tabulate*)
  - gives a crosstab of *x1* vs *x2*
  - use only if *x1* and *x2* are integers

Tabulating

- *tab x1 x2 if x4==0, sum(x3)*
  - gives the means of *x3* for each cell of the *x1* vs *x2* crosstab for obs where *x4*=0 (note ==)
- *tab x1 x2, missing*
  - Includes the missing values
- *tab x1 x2, nolabel*
  - Uses numeric codes instead of labels
- *tab x1 x2, col*
  - Gives % of column instead of count
- *table degree71 ethnic, c(mean age) row col*
  - Customises the table

Data manipulation

- Data can be renamed, recoded, and transformed:

gen logwgr=log(grsswk)

gen agesq=age^2 ^ raises to the power

gen id = _n *_n* is obs # in STATA

gen ylagged=y[_n-1]

replace rate=rate/100 *rescale the var*

replace age=25 if age==250

rename agesq agesquare

Sorting data

- *sort* and *count*
  - *sort id*
    - sorts data by person id
  - *count if id == id[_n-1]*
    - counts how many unique separate personids
    - *_n-1* is the previous observation
  - *duplicates report id*
    - *same function count if*

Extended generate (*egen*)

egen

- Useful when you need a new variable that is the mean, median, etc. of another variable
  - for all observations or for groups of observations.
- *A*lso useful when you need to simply number groups of observations based on some classification variables.
- Great when you have panel data

egen examples

egen sumvar1 = sum(var1)

creates sumvar1 as sum of values of var1

egen meanvar1= mean(var1), by(var3)

creates meanvar1 as mean of all values of var1

egen counter = count(id), by(company)

creates count as the number of companies with nonmissingid's

egen groupid = group(month year)

assigns a number to each month/year group

Handling string variables

encode

- Use *encode* when the original var is a character var (e.g. *gender* is "*m*" or "*f*")
- *encode* command does not produce dummy variables, it just assigns numbers to each group defined by the character variable.
- In this example, *gender* was the original character var and *sex* is new numeric var:

  encode gender, gen(sex)

- *decode* does the opposite
- destring

Dummy variables

- *gen wales_resid=0*

  replace wales_resid=1 if uresmc==17
- *gen region17=(uresmc==17)*
- to create a series of dummies from a categorical var
  - *tab uresmc, gen(dresid)*
  - xi: sum i.uresmc
- *recode uresmc (1/16=1) (18 19=2) (17=3) (20=4), gen(state_resid)*
- *gen engwales_resid= 0 if state_resid==3*
- *replace engwales_resid=1 if state_resid==1*

Labelling

- Always a good idea to have your data comprehensively labelled

  label data "LFS third quarter 2008"

  label var state_resid "residence by country"

  lab values state_resid state_resid

  lab def state_resid 1 "england" 2 "scotland" 3 "wales" 4 "north ireland", modify *cmd in 1 row*

- Tedious to do for lots of variables
  - but then your output will be intelligibly labelled
  - other people will be able to understand

Using STATA as a calculator

- *display* command
  - *dis 22/7*
  - *disp log(250)*
  - *di exp(3.6)*
  - *di chiprob(2,6.45)* (i.e. 2 df, deviance 6.45)
    - returns 0.398 (i.e. its significant at 5% level)
  - *display _N*
    - Returns the sample size
      - (*_N* is the number of the last obs)

Data selection

- Organise your data with various commands:

  keep if _n<=1000 *_n* is the obs number

  drop x1

  drop if x2 ~=1

  keeps only the first 1000 obs, drops *x1*, and drops all the observations where the variable *x2≠1* (~= is "not equal to")

Syntax to remember

>= means "greater or equal",
& means "and",
| means "or"
= means “set equal to”
== means “is it equal to?”
~= means “not equal” (or use **!=**)
. means missing value

For example

keep if x1>= 1 & x2<=3 | x2==7 & x3 ~= .

Functions

- Lots of functions are possible.
- See *help functions*
  - Obvious ones like
    - *log(), abs(), int(), round(), sqrt(), min(), max(), sum()*
  - And many very specialised ones.
  - Statistical functions
    - distributions
  - String functions
    - Converting strings to numbers and vice versa
  - Date functions
    - Converting dates to numbers and vice versa
  - And lots more

Merging data - 1

- file1 has *id x1 x2 x3* , file2 has *id x4 x4 x5.*
- You can merge using "key" in BOTH files (*id*)
- But you need to *sort* both files first.

```
use file1
gen id =_n                  if personid doesn't exist already
sort id                sorts file1 according to id variable
save, replace
use file 2
gen id =_n
sort id                sorts file2 according to id variable
merge id using file1
drop if  _merge~=3     drops obs with any missing info
save file3
```

Merging data - 2

- For each row (*id*) all vars in *file1* added to corresponding row of *file2* (if there is one).
- *merge* creates a new variable, *_merge*
  - which =1 for those obs only in *file1*, =2 for those only in *file2*, and =3 for those in **both**.
- So the syntax above drops those obs that don't have data in both files
  - and saves the result containing x1-x6 in *file3*
- *append* to add more obs on the same vars.

Collapsing data (use with care)

- collapse converts the data in memory into a dataset of means (or sums, medians, etc.)
- This is useful when you want to provide summary info at a higher level of aggregation
  - For example, suppose a dataset contains data on individuals – say their region and whether u/e
  - To find the average u/e rates across reg type:

    collapse unemp, by(region)

  leaves 1 obs for each reg and mean u/e rate.

Reshaping files

- Data may be "long" but thin
  - Eg each record is a household member
  - But there are few vars - say *wage* and *hours*
- Data may be "wide" but short
  - each record is a household and has lots of vars
  - (eg *w1 w2 w3 hours1 hours2 hours3*)

reshape long inc ue, i(id) j(year) wide to long

reshape wide inc ue, i(id) j(year) back to wide

Handy for merging data together and for panel data

Some useful websites

- Stata's own resources for learning STATA
  - Stata website, Stata library, Statalist archive
  - http://www.stata.com/links/resources1.html
- Michigan's web-based guide (for SA)
- UCLA resources
  - http://www. ats.ucla.edu/stat/ stata/
- ESDS "Stata for LFS"
- Princeton; Illinois; Gruhn